Clouds and the Earth's Radiant Energy System (CERES)

Data Management System

Software Design Document

Instrument Geolocate and
Calibrate Earth Radiances
(Subsystem 1.0)

Architectural Draft

Troy Anselmo, Denise Cooper, Joey Escuadra,
Anna Estes, Phil Hess, Alice Lake, and Larry Matthias

Science Applications International Corporation (SAIC)
One Enterprise Parkway, Suite 300
Hampton, Virginia 23666

Atmospheric Sciences Division
NASA Langley Research Center
Hampton, Virginia 23681-0001

Release 1.0
June 1996

Preface

The Clouds and the Earth's Radiant Energy System (CERES) Data Management System supports the data processing needs of the CERES science research to increase understanding of the Earth's climate and radiant environment. The CERES Data Management Team works with the CERES Science Team to develop the software necessary to support the science algorithms. This software, being developed to operate at the Langley Distributed Active Archive Center (DAAC), produces an extensive set of science data products.

The Data Management System consists of 12 subsystems; each subsystem represents a stand-alone executable program. Each subsystem executes when all of its required input data sets are available and produces one or more archival science products.

The documentation for each subsystem describes the software design at various stages of the development process and includes items such as Software Requirements Documents, Data Products Catalogs, Software Design Documents, Software Test Plans, and User's Guides.

This version of the Software Design Document records the architectural design of each Subsystem for Release 1 code development and testing of the CERES science algorithms. This is a PRELIMINARY document, intended for internal distribution only. Its primary purpose is to record what was done to accomplish Release 1 development and to be used as a reference for Release 2 development.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 Introduction

The Clouds and the Earth’s Radiant Energy System (CERES) is a key component of the Earth Observing System (EOS). The CERES instruments are improved models of the Earth Radiation Budget Experiment (ERBE) scanner instruments, which operated from 1984 through 1990 on the National Aeronautics and Space Administration’s (NASAs) Earth Radiation Budget Satellite (ERBS) and on the National Oceanic and Atmospheric Administration’s (NOAAs) operational weather satellites NOAA-9 and NOAA-10.

The strategy of flying instruments on Sun-synchronous, polar orbiting satellites, such as NOAA-9 and NOAA-10, simultaneously with instruments on satellites that have precessing orbits in lower inclinations, such as ERBS, was successfully developed by ERBE to reduce time sampling errors. CERES will continue that strategy by flying instruments on the polar orbiting EOS platforms simultaneously with an instrument on the Tropical Rainfall Measuring Mission (TRMM) spacecraft, which has an orbital inclination of 35 degrees.

In addition, to reduce the uncertainty in data interpretation and to improve the consistency between the cloud parameters and the radiation field parameters, CERES will include cloud imager data and other atmospheric parameters during data processing. The first CERES instrument is scheduled to be launched on the TRMM spacecraft in 1997. Additional CERES instruments will fly on the EOS-AM platforms, the first of which is scheduled for launch in 1998, and on the EOS-PM platforms, the first of which is scheduled for launch in 2000.

1.1 Document Overview

The purpose of this document is to present and describe the architectural software design for the Geolocate and Calibrate Earth Radiance, Instrument Processing Subsystem (IPS). The objective of this document is to provide readers with a technical understanding of the IPS architecture which forms the basis for developing: interface designs and specifications, detailed designs, code, maintenance specifications, and test strategies and procedures.

This document is intended for audiences ranging from readers who want a high level background informational summary of the software’s core functionality, to readers who may be software developers or programmers needing to understand technical design and implementation consideration aspects of the software.

As a living document, this document is planned to be updated on an irregular basis. Updates are expected in response to changes in Subsystem functional requirements, software design, coding implementations, or testing results. The frequencies, occurrences, and types of updates to this document will depend primarily on the scope or magnitude of the changes effected.

The scope of this document includes discussions of a subsystem overview, key concepts, system design drivers, assumption and trade-off considerations, key interfaces, and the architectural design. The architectural design presentation includes both "static" (interfaces) and "dynamic" (processing flow) views of the Subsystem software. The Appendices at the end of this document contain additional information about specific topics which may be of further interest to the reader. Examples to be anticipated in the Release 2 version of the design document include a Requirements Matrix and Error Messages.

Not included in this document is information about software detailed designs, software development strategies (e.g., waterfall, incremental builds, etc.), testing strategies and procedures, and quality assurance plans.

This architectural design is based on and derived from the CERES Algorithm Theoretical Basis Document (ATBD), Data Management System (DMS) Software Requirements Document (SRD), DMS Interface Requirements Document (IRD), DMS Data Products Catalog (DPC), Specification of Algorithms to Calculate Geolocations of the CERES Instrument Radiance Measurements and other Earth-Sun-Spacecraft Parameters for the CERES DMS, CERES Instrument Operations Manual, and the CERES In-flight Measurement Analysis Document (References 1 through 7).

The document outline is shown below:

- 1.0 Introduction
    - 1.1 Document Overview
    - 1.2 Subsystem Overview
    - 1.3 Key Concepts
    - 1.4 Design Goals
    - 1.5 Design Considerations, Assumptions, and Trade-Offs
    - 1.6 Implementation Constraints
    - 1.7 Design Approach
- 2.0 Architectural Design
    - 2.1 Class Diagrams
    - 2.2 Scenario Diagrams
- References
- Appendix A - Abbreviations and Acronyms

The following sections are not included in this release:

- 3.0 Detailed Design (Module Specifications)
- Appendix - Requirements Matrix
- Appendix - Test Performance Report Format
- Appendix - External Interfaces
- Appendix - Data and Constants
- Appendix - Resources
- Appendix - Error Messages
- Appendix - Booches Object Oriented Reference

1.2 Subsystem Overview

The IPS is the first Subsystem in the CERES Data Management System (DMS). The primary purpose of the IPS is to process raw spacecraft and instrument sensor and engineering telemetry data into output geolocated radiance data products for subsequent CERES DMS processing. Specifically, this processing can be broken down into the following three major functions:

1. Convert raw instrument:
    a) sensor outputs (counts) into filtered radiance values and,
    b) analog and digital engineering data into engineering units.
2. Geophysically locate field-of-view (FOV) data measurements.
3. Perform Quality Control (QC), Quality Assurance (QA), and data validation checks to ensure the integrity and quality of the IPS data output products.

The Subsystem interfaces associated with this processing is depicted graphically by the Release 1 Context Diagram (Figure 1-1). Associated detailed overview information can be found in the Software Requirements Document (Reference 2).

The primary data input for the IPS is called a level-0 file. This file is actually several physical files, but they are represented as a single virtual file by the EOSDIS Core System (ECS) Toolkit. The level-0 file contains chronologically ordered, by data type Application Identifier (APID), instrument data packets. These packets are expected to have been reconstituted and decommutated from the spacecraft-to-ground transmission links. It is anticipated that unrecoverable "bad" data will be purged from this file and an accounting of resulting data gaps will be included with this file. The specific level-0 file format is dictated by the TRMM or EOS-to-EOSDIS project interface agreements. Since CERES does not produce a Level-1A data product, the EOSDIS is expected to archive Level-0 data.

Within the level-0 file, each packet contains instrument data collected during a single 6.6-second interval. This typically corresponds to an instrument elevation scan period (normal operation). The format of these telemetry packets conforms to the Consultative Committee for Space Data Systems (CCSDS) communication protocol. This protocol provides for a header in addition to the primary instrument detector and engineering output data.

Under most conditions, a typical level-0 file will contain 24 hours of instrument detector and engineering data, corresponding to approximately 13,091 packets. The first packet typically begins at or after midnight Universal time (>00:00:00.0) and the last full packet typically ends with data transitioning the end of the day (@23:59:59.999999Z).

Other input files are required or needed to process this level-0 file. These files support data conversion, geolocation, radiance evaluations, and QA/validation functions. Examples of secondary input or ancillary files include the corresponding daily ephemeris data file, instrument parameter conversion coefficients file, and calibration coefficient history file.

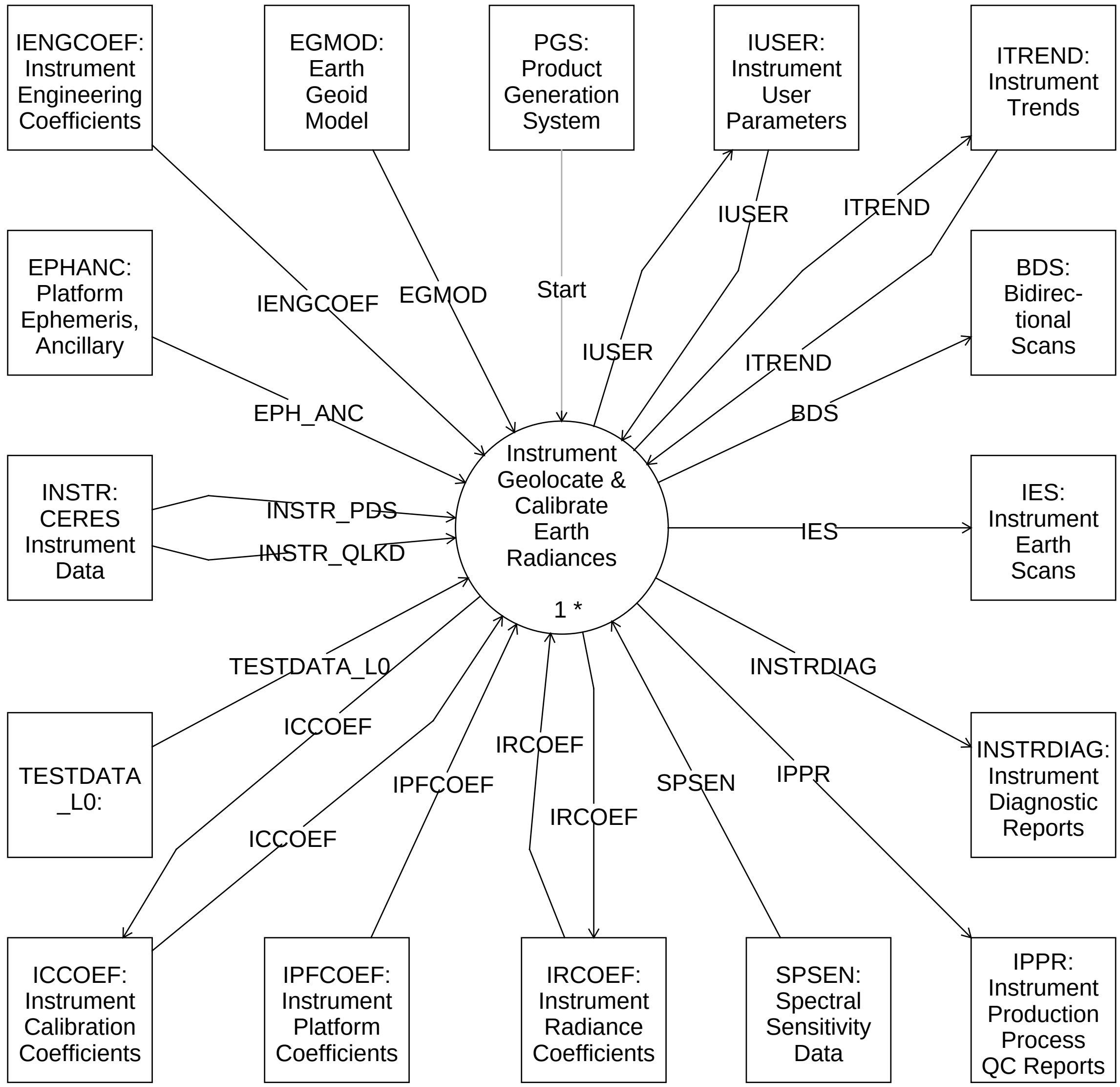

Figure 1-1. Context Diagram

The primary IPS output data results are two science data products: A daily BiDirectional Scan (BDS) file and hourly Instrument Earth Scan (IES) files. The BDS file is an archival product which contain up to 24 hours of data that correspond to the level-0 input file. This includes all raw analog and digital instrument data from the level-0 file, the corresponding converted values (radiances and engineering units), ephemeris and geolocation parameters, and associated quality data and processing flags. The specific data parameters contained within the BDS are defined in the CERES Data Products Catalog (Reference 4).

The BDS file primarily serves as the input product for the CERES Data Management System (DMS) ERBE-like Subsystem (SS 2.0); it will also serve as an input product for the IPS in reprocessing scenarios. In addition, the BDS file will serve as the data set used for anticipated off-line engineering and science validation efforts. If needed, future data reprocessing will use the BDS file as the primary IPS data input (since the original level-0 files are not expected to be archived). The data format is expected to be dictated by agreements between the IPS and data users in conformance with the EOSDIS mandated Hierarchical Data Format (HDF) protocol.

The IES output data products are a collection of twenty-four 1-hour data files which normally correspond to the level-0 file data. The specific data parameters contained in an IES file are defined in the CERES DMS Data Products Catalog. Unlike the BDS file, IES files are considered internal CERES DMS data products and are not archived. IES files do not contain raw or converted instrument sensor or engineering data. The primary data elements in an IES file are geolocated radiance values which are sorted temporally and spatially by data subset units called footprints. This sorting of footprint data is necessary to support DMS Clouds Subsystem(s) (SS 4.1-4.6) processing.

1.3 Key Concepts

The following key concepts are embodied within the Architectural Design of the Instrument Geolocate and Calibrate Earth Radiances Subsystem.

- *Input Data Organization and Access*
- *Instrument Performance Evaluation*
- *Geolocation Calculations*
- *Space Clamp and Radiance Processing*
- *Calibration Processing*
- *Output Product Considerations*
- *Processing Synchronization*
- *Production and Processing Environment*
- *Metadata and Error Handling*
- *Data Quality*

Input Data Organization and Access: The primary input instrument data are accessed as level-0 files retrieved by mandatory level-0 Toolkit routines. Level-0 data are contained in three distinct data files that are organized by Application Identifiers (APIDs). The APIDs correspond to science, solar calibration, and diagnostic data. Each data file can contain up to 24 hours of data. However, the combination of the three data files must add up to 24 hours of data.

Science APID data consist primarily of all normal "science" (radiance and scanner position) and "nonscience" engineering data measured within the instrument’s fixed or rotating azimuth, normal Earth-scan configuration. Solar calibration APID data consist of the same data measured during a fixed azimuth, Sun-viewing Mirror Attenuator Mosaic (MAM) scan configuration. Diagnostic APID consist of all data measured within other instrument configurations (e.g., memory dumps and science data that are for evaluation purposes but are not expected to be archived).

All CERES instrument data within these level-0 files are organized into CCSDS formatted data packets containing 6.6 seconds of instrument measurements. Each packet contains 660 data records of science, calibration, or instrument diagnostic measurements with associated instrument digital status and analog engineering measurements. The data records are formatted to either science or diagnostic packet formats. There is only one science data format, but there are several types of diagnostic data formats. These data formats are for memory, gimbal error, processor, and fixed pattern data.

Furthermore, these telemetry packet data are typically formatted as packed data. A 16-bit word can be made up of one-to-N parameters in m-bit chunks. It is highly desirable to unpack these data in a single module and serve up all parameters in easier to handle byte, word, or long-word sizes. This will provide easier maintainability and portability by reducing the impact of potentially inconsistent machine or language implementations.

Additional input data required for processing include the associated 24-hour Ephemeris data file and supporting ancillary files. Ancillary files will be developed off-line and are shown in Figure 1-1.

Instrument Performance Evaluation: While geolocated science radiances are the primary data output of interest to the user community, the quality of these results is heavily dependent on successfully verifying and validating the instrument's operating performance (References 6 and 7). Instrument performance involves a comprehensive evaluation of all digital status and analog measurements and their interrelationships (e.g., detector, temperature, power, and position measurements).

Evaluations involve limit (edit) checks; rate of change checks; trending limit checks; parameter validation; statistical averaging and deviation analyses; and fault isolation, verification, and validation. Instrument configuration and command verification, logic tree comparisons, and historical tracking is expected to be implemented. Instrument operations require analysis of the command structure within normal packets and diagnostic packet processor operations, memory dumps, and gimbal error output data. Instrument diagnostic data evaluations will determine data processing changes that may be required. Also, to support rate edit check and associated statistical processing, it is desired to have immediately available both the current and previous chronological packet. Packets will be checked for proper time sequence, and data gaps will be flagged.

The design of the processing software closely mirrors the partitioning of the CERES instrument mechanical and electronic subsystems. For example, software modules are dedicated exclusively to processing data from the Azimuth Gimbal Assembly, Elevation Gimbal Assembly, and the Detector Assembly. This mirroring closely corresponds to the flight software and makes operation and anomaly investigation easier to comprehend.

Geolocation Calculations: To properly interpret the CERES radiance measurements, it is essential that accurate Earth fields-of-view be determined for each instrument measurement (Reference 5). This is accomplished by knowing the instrument detector elevation and azimuth angles for each measurement, the corresponding spacecraft ephemeris (sample time, position, velocity, and attitude), and the various vector coordinate system geometries. Earth locations are calculated by a series of coordinate transformations into unit pointing vectors, then determining a pierce point for both the Top-of-Atmosphere (TOA) and surface geoid. Various pointing errors are considered in the computation process and must include timing accuracy of instrument measurements and spacecraft-to-detector measurements to Universal time, definitive spacecraft position accuracy, and detector-to-instrument and instrument-to-spacecraft mechanical alignments.

With the advent of the ECS Toolkit, many of these calculations will be performed by calling common Toolkit functions. The starting point for these functions begins with a given FOV measurement vector having been translated to the spacecraft coordinate system. Therefore, CERES must perform the calculations and verifications to translate azimuth and elevation angles to the spacecraft coordinate frame.

Space Clamp and Radiance Processing: Due to historical sensor performance variabilities and a current more sensitive, yet unproven sensor, radiance processing designs need to accommodate flexible algorithm options. Space Clamp algorithm options based on different space look regions within and across many scans are expected. Designs for radiance count conversion algorithms need to accommodate several options dictated by the CERES Science Team. Options include coefficient updating criteria and frequency, within scan or multiscan parameter evaluations, and corrections for the second time constant.

To support desired space clamp selection options (Calibration Concepts document (Reference 8)), it will be necessary to have immediately available both the current and next packet. The next packet is assumed to be a chronologically sequential packet (i.e., no time gapped data). However, this processing must allow for missing packets.

Calibration Processing: To ensure meaningful science quality, accurate knowledge of the radiance detection and measurement process is required and must be incorporated into the Subsystem design. This knowledge is acquired via ground and in-flight calibration processes. Ground calibrations provide initial characterization data for the three instrument radiance detector sensors and the internal calibration sources. These data provide characterizations as a function of temperatures, voltages, and external "standard" calibrated measurement references. These data are used during production processing to verify and evaluate in-flight calibration and radiance detector performance.

In-flight calibrations use solar and internal calibration sources. Solar viewing Mirror Attenuator Mosaics (MAMs) are used for calibrating the shortwave and total detector channels. Solar calibrations are expected to be performed approximately every two weeks throughout the mission. This calibration procedure requires a special elevation scan profile to accommodate MAM views. Internal calibration sources include blackbodies for calibrating total and longwave (window) channels, and a Shortwave Internal Calibration Source (SWICS) for calibrating the shortwave channel. The internal calibration sources (blackbodies and SWICS) are viewed each science scan, although they are normally not active. These sources will be activated normally during scheduled internal calibration events only. In addition, it is anticipated that the platform(s) will perform a special on-orbit maneuver sometime during the mission to allow the instrument to view deep space, thereby identifying scanner position dependent influences.

Calibration data will be used for verifying, validating, and possibly adjusting the radiance conversion coefficients based on in-flight calibration processes. The radiance conversion coefficients are expected to be updated periodically throughout the mission to accommodate natural instrument degradation effects. The update process is expected to follow EOSDIS procedures being defined as the EOSDIS System evolves. The suggested concept is to flag in-flight calibration data that has exceeded predefined criteria for review by the CERES Science Team. Upon evaluation and approval, conversion coefficients may be updated in the production environment.

However, it is planned that data analyst procedures performed off-line during the ERBE mission, will be automated as part of on-line processing. These include expanded trending evaluations, on-line plotting, and fault isolation. Plotting capabilities will be automatically invoked by anomalous production and instrument data conditions. Optional feedback mechanisms should be considered as part of the architectural design.

Output Product Considerations: Output products are determined and driven by two major considerations - Subsystem interface data structure requirements and instrument validation efforts. Interface requirements are dictated by and coordinated with Subsystems 2.0 (ERBE-like Inversion) and 4.4 (Convolution of Cloud Properties). Both of these Subsystems require level-1b radiance data. These interface data products are categorized as archival or intermediate data. Instrument validation efforts generate comprehensive processing reports and statistical data that will be archived. A common output concept is the "footprint," which is a multiband structure containing radiance, geolocation, and quality information associated with one instrument sampling period. This structure is called a footprint to differentiate it from the "pixel" organization of the cloud imager products used in subsequent Subsystems. Actual file physical formats will be driven by HDF and EOS-HDF requirements and specifications. The output products are summarized below.

1. The BiDirectional Scan (BDS) data organization reflects ERBE heritage and the downstream ERBE-like processing consistency requirements. These data are organized into 24-hour files with 6.6-second records. This output product also provides two additional capabilities. These are:

   a) Satisfy the EOSDIS requirement for being able to "reconstruct" level-0 data from level-1b data. The BDS will contain a decommutated, unpacked copy of the level-0 data to accomplish this.

   b) Provide a full set of converted instrument engineering data to aid in instrument anomaly investigations.

2. The Instrument Earth Scan (IES) data organization reflects the need to coordinate with cloud imager data that are processed in the CERES-unique Subsystems. These data are

organized into 1-hour files containing spatially ordered footprints. No engineering data are included in the IES file.

3. Instrument validation outputs include all engineering statistical analyses, trending parameters, and processing quality control reports.

Processing Synchronization: The overall IPS processing scenario is to process daily files and packets sequentially and synchronously. During normal production processing, level-0 files will be staged and processed in chronological daily order to support IES spatial sorting requirements involving overlapping footprints from the previous day. Within each level-0 file, the packets are read in a time-ordered sequence and processed one at a time. This single packet read and process sequence will maintain processing synchronization among the instrument, radiance, geolocation, QC, and output data writing subprocesses. Processing sequential packets will be accomplished regardless of the file source (science, calibration, or diagnostic) or the scan profile type (normal-Earth, short-Earth, or MAM scans).

It is also required that at the beginning of each production run (i.e., when the Subsystem Controller Module is executed), an initialization subprocess will be invoked. This initialization process will be required before any packets can be processed. The initialization will, among other things, access and open all relevant files; read in ancillary files needed for processing; initialize global variables (including metadata information); create and open required output files; and if all is successful, get the first packet and verify the level-0 file header and footer.

Production and Processing Environment: The Subsystem code will conform to EOSDIS concepts for supporting staging and ingesting data, using system resources, incorporating operating protocols, and insuring timely results. Detailed impacts on the code development have been deferred until more information is available from EOSDIS. In the interim, some assumptions about the processing environment have been made from the ECS Operations Concept Document (Reference 9). These include the following:

1. File header verification and validation is anticipated to be handled by the system; however, the IPS software will also perform this function.
2. Data staging and code execution will be performed by Product Generation Executive (PGE) scripts.
3. System messages will be generated by the Subsystem using ECS toolkit functions.
4. Operating guidelines will be established for all IPS processing scenarios.
5. Runtime user information (e.g., environment variables) may need to be passed into the program via the Process Control File.

Metadata and Error Handling: Metadata and error handling will conform to the EOSDIS operating environments, as specified in the Toolkit Version 5.1, Release 2 issue (Reference 10). Details regarding metadata have been deferred until more information on the operating environment is available.

Data Quality: All bad level-0 data packets due to EOSDIS transmission processes are expected to have been eliminated from the input data stream before a Subsystem PGE is executed. Examples of bad data include packet dropouts or incorrectly reconstituted data via the Tracking and Data Relay Satellite System (TRDSS) error correction algorithms. All bad level-1 generated BDS and IES data will either be flagged as bad (BDS) or will not be output (IES). A CERES standard fill value (e.g., -9999.999) will be written for out-of-limit parameters.

1.4 Design Goals

1. Process 1 hour of level-0 data in an elapsed time of one hour or less. It is desired that this objective be met while including validation, verification, diagnostic, and/or debugging processing.
2. Develop modules to minimize data coupling (i.e., the number of modules impacted should parameter changes be required). For example, provide separate data definition modules mapped to the problem domain such as geolocation, radiance, engineering, or mathematical types.
3. Reuse wherever possible public Ada library or generic code.

1.5 Design Considerations, Assumptions, and Trade-offs

Design considerations, trade-offs, and assumptions ("drivers") that are, or need to be, captured in the architectural and detailed design are included below. (Some of these topics are covered in more detailed in the Implementation Constraints Section.)

1. System Level:
   a) DAAC/Science Computing Facility (SCF) Drivers: Within the EOSDIS environment, processing requirements, interfaces, and resource limitations are evident. These include processing turnaround times, delivery procedures, configuration management, memory and storage usage and limitations, and operating protocol (e.g., PGE scripting rules).

   b) Instrument Drivers: Subsystem design and processing considerations are heavily driven by mission operational configurations; instrument variations, both known and anticipated (e.g., ERBE operational variations); science usages; and data makeups and procedures.

   c) Processing Algorithm Drivers: Algorithm coordination, sequencing, data sources, and resource needs must be considered for geolocation, radiance, space clamp, IES sorting, and statistical summaries.

   d) Key Interfaces: External file protocols and formats (e.g., HDF) should address production, validation, and off-line user needs (e.g., solar calibration data).

   e) Data Sources and Formats: File formats and data organization need to be addressed for any ancillary files and output data products as directed by users.

2. Processing Environment:

   a) Scheduling and Data Accesses: These access requirements, protocols, and scheduling need to be evaluated for any process flow influences.

   b) Memory: DAAC/SCF and development machine capacities and utilization techniques may influence buffer configuration and usage designs.

3. ToolKit:

   a) Operational Drivers: Underlying data sources, environments, and operating performance may influence interface designs or calling protocols.

   b) Language Bindings: Toolkit routines are predominately written in C. Careful considerations must be given to matching syntax and semantics between different languages. Isolating binding calls to separate modules can smooth integration, modifications, and portability.

4. Problem Domain Considerations:

   a) Instrument processing sequences: Engineering and status data conversions and state evaluations should reflect actual operating sequences and relationships. For example, status data reflects operation and should be evaluated independently before command sequence comparison.

   b) Radiance and Calibration processing: ERBE history demonstrated the need for flexibility in adapting radiance, space clamp, and calibration processing to accommodate algorithm adjustments and options. Of concern, is identifying and eliminating instrument influences on the count conversion process and verifying calibration source versus detector discrepancies.

   c) Software Development Methodology and Language Drivers: It is desired to develop an architecture that minimizes the influences of implementation languages. However, development paradigms may influence some design aspects (e.g., bindings, environment variable inclusions, library interactions, mathematical accuracy requirements).

5. Processing Functions Tradeoffs: Design tradeoffs are expected to be addressed among known processing, requirement, processing protocol, user, algorithm concept, targeted platform, operating system, and compiler variations. The following functions can influence these processing tradeoffs.

   a) Controller module level of intelligence

   b) Space clamp algorithm options

   c) Calibration algorithm options

   d) Dynamic memory management

   e) Packet-to-other module relationships

   f) Multiple scan buffering

g) Data flow through efficiencies

h) Error exception handling

i) Module include (with) strategy

j) Quicklook processing

1.6 Implementation Constraints

The following section highlights design requirements or constraints that affect the analysis, design, and development of the IPS software. Some of these factors are listed and described below.

- *Programming Language*
- *Data Sizes*
- *Design and Development Approach*
- *Dynamic Memory*
- *ECS Toolkit*
- *Hierarchical Data Format (HDF)*
- *Ada Quality and Style: Guidelines for Professional Programmers*

Programming Language: Prior to the design and code development phase, Ada was selected as the programming language for the IPS. This is in contrast to other DMS subsystems which are using FORTRAN 77/90. As a result, some additional effort is required to design and implement interfaces to mandatory libraries (see ECS Toolkit and HDF discussion), as these libraries are typically implemented in other languages.

Data Sizes: For efficient spacecraft-to-ground transmission, telemetry data streams are typically formatted as packed data. A given 16-bit telemetry word may represent a single parameter or could be made up of multiple parameters in smaller n-bit representations. Care must be given to recognizing language and platform limitations when unpacking the data. Considerations include big-endian/little-endian, bit manipulation techniques, and byte ordering.

Design and Development Approach: The IPS is being designed and developed using an Object Oriented Design and Analysis (OODA) approach. This approach is reflected in documentation descriptions of the IPS software and in the design and implementation of the IPS code and is based on Booch's object oriented design methodology and notation. (Comprehensive information on the Booch methodology is planned to be included in the next release of this document.) The intent of this approach is to develop and provide a system supporting a long (15-20 year) production processing environment with minimal operating costs.

Dynamic Memory: Consideration must be given to the use of dynamic memory within the system. Factors to be considered include availability within the development and target hardware platforms, restrictions imposed within the ECS processing environment, and restrictions resulting from the development language and tools (compilers, debuggers, etc.).

ECS Toolkit: Use of certain ECS Toolkit library functions is mandatory for all of CERES DMS subsystems. As a minimum, the IPS will use the Toolkit library to handle opening and reading of level-0 files, retrieve runtime parameters, and log system and instrument code error and status messages. The IPS will also use the Toolkit library to perform time operations, compute instrument Field-of-View (FOV) geolocations, and compute satellite/celestial geolocations. Use of memory management tools is to be determined. The Toolkit library is primarily written in C, but also contains extensive FORTRAN 77 code.

Hierarchical Data Format (HDF): HDF was chosen by the EOSDIS as the primary output data product storage format that all DMS subsystems will use. For IPS, the BDS file and IES files are to be implemented in HDF. HDF supports several standardized data formats and provides Application Programming Interface (API) libraries for each of these formats. These libraries are written in C and FORTRAN 77. Formats and subdata sets must be defined in the detailed design phase and should be consistent with architectural design considerations.

Ada Quality and Style: Guidelines for Professional Programmers: The intent of the IPS architectural design is to minimize software language related dependency. The current plan is to develop the IPS code using Ada. Therefore, the Ada Quality and Style document (Reference 11) provides established guidelines and conventions for writing Ada code and shall be adopted for all IPS code development.

1.7 Design Approach

It is planned that the architectural design approach be developed using the OODA based paradigm. The goal of this paradigm is to develop a design that closely maps to the problem domain. This mapping process will help users and programmers to understand the software and its development, to provide better and efficient long-term maintainability and portability, and to create resilient and robust code.

The essence of this paradigm is to create objects (modules) incorporating problem space states, behaviors, and identities as independent entities. This is accomplished by combining both data (domain states) and functions (behavior operations) within potentially logical abstractions (i.e., modules). The procedure for identifying such abstracted objects involves developing two types of diagrams that are then iterated. The diagrams involve static (class) and dynamic (scenario) representative solutions to meet the processing objectives. Diagrams are evaluated for commonality, interface effects, processing consistency, and common sense. Adjustments are made to achieve better cohesiveness among classes (modules) and better interface decoupling. These adjustments are iterated until either all requirements (previously defined) are met or understanding limits are reached. Prototyping is expected to test solutions to resolve these limits and allow the iteration process to continue.

Within the IPS, high-level objects (classes) were developed based on specific categorized processes. The primary classes are illustrated in the Section 2.1 class diagrams.

This design approach led to the following architectural features:

1. The overall Subsystem architecture is divided into four functional areas. These areas are input (e.g., level-0 IO, packet); instrument engineering (e.g., instrument); science (e.g., footprint, location, radiance); and output processing (e.g., BDS, IES).
2. System initialization will be done once at the beginning of a PGS execution. All subprocesses requiring initialization will be invoked. Subprocesses requiring ancillary files will require accesses and verifications.
3. The reading of level-0 input files and the writing of the BDS and IES output products are handled by separate classes.
4. Level-0 data packets are processed one at a time.
5. Instrument engineering data are processed and QA’d before science data.
6. The IES module will identify hourly time boundaries and provide spatial sorting processes. IES will store approximately one hour of data before sorting and then write that hour to an IES file.
7. The BDS module will store one scan of data and write each scan of data before Controller calls subsequent packets.

2.0 Architectural Design

The IPS architectural design presented in this document will be described in terms of Class Diagrams and Scenario Diagrams. Class diagrams can be thought of as "collections" of data structures, functions that operate on these data sets, and representative module interfaces. The class diagrams can sometimes be referred to as "static" portrayals of the design. Scenario diagrams can be thought of as representing module data flows or processing sequences. These scenario diagrams are sometimes referred to as "dynamic" portrayals of the design.

2.1 Class Diagrams

The top level IPS class diagram is shown in Figure 2-1. This diagram uses the Booch design methodology and notation. A further breakdown of the *Instrument Configuration* class yields the classes shown in Figure 2-2. Together, the diagrams of Figure 2-1 and Figure 2-2 represent an abstract analysis of the science and instrument processing problem domain.

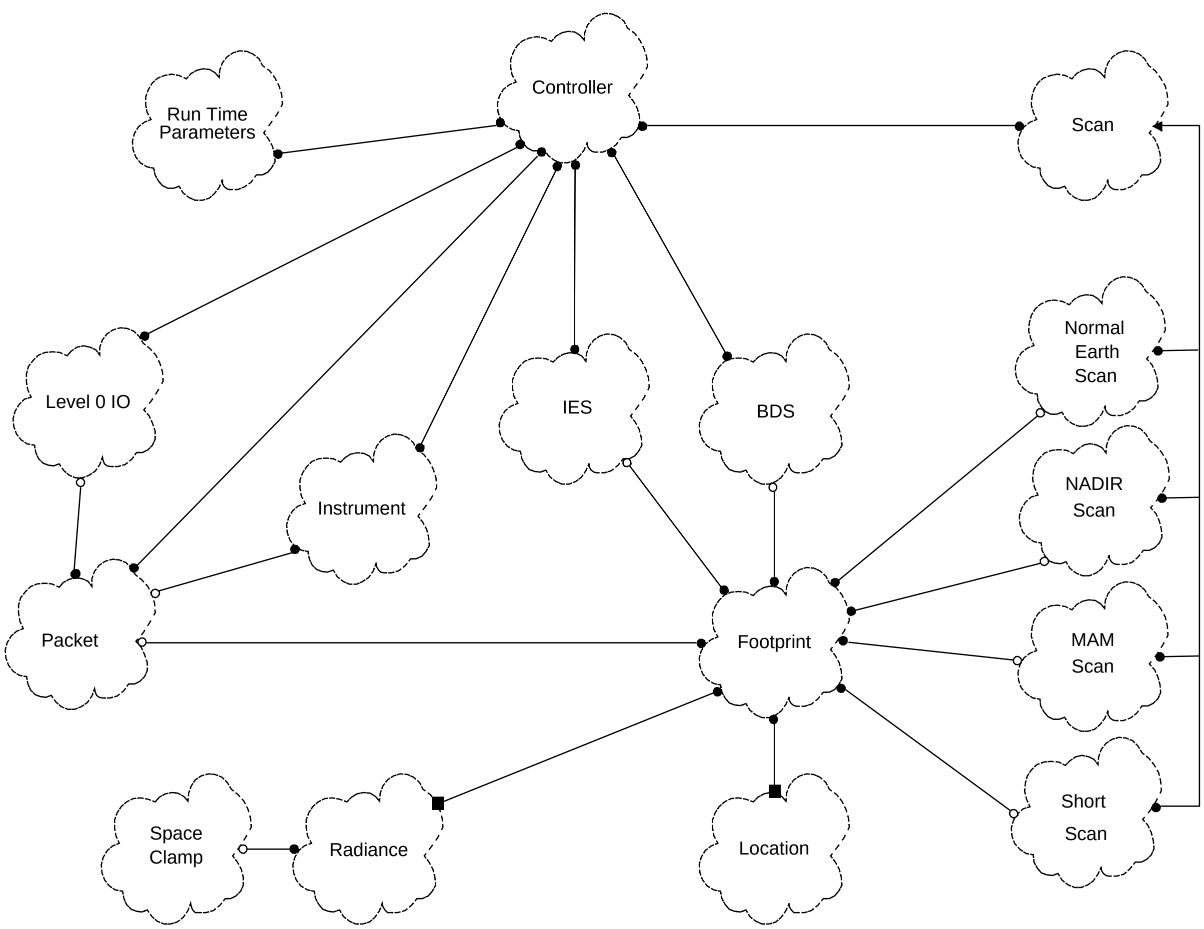

Figure 2-1. Class Diagram: IPS Architecture

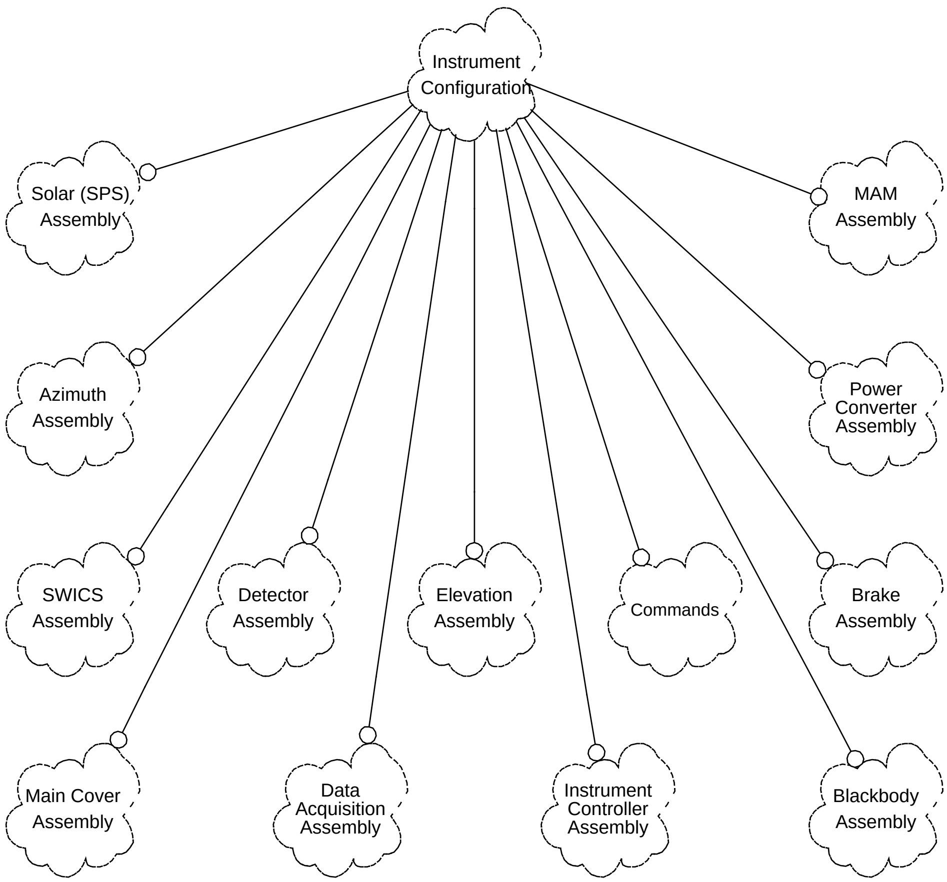

Figure 2-2. Class Diagram: Instrument Configuration Classes

Each class plays an integral part in meeting the IPS system level functional requirements as stated in the Subsystem Overview (Section 1.2) and other requirements documents. The lines in the diagrams connecting different classes depict relationships that exists between the classes. Discussions of Toolkit functions will be addressed in the appropriate classes where the Toolkit calls are made.

Descriptions of each of these classes are provided below.

Controller: The initiator and master processing coordinator of the IPS, *Controller* coordinates data flow and message traffic between the related connecting classes. *Controller* is also responsible for gracefully aborting the IPS processing should a fatal exception occur in any of the other classes.

Level-0 IO: This class contains the data elements and processing procedures required to open and access a level-0 file. In addition to opening and closing the level-0 file, *Level-0 IO* is also responsible for checking and verifying the file header(s) and footer(s), passing packet data blocks to *Packet*, and returning its processing status to *Controller*. *Level-0 IO* could cause *Controller* to halt IPS processing if a level-0 file cannot be opened or read. *Level-0 IO* uses the mandatory level-0, PCF, Time, and System Management Function (SMF) Toolkit routines.

Packet: This class processes data from the level-0 file that contains a single scan of detector and instrument housekeeping data. In *Packet*, the generic block of data from *Level-0 IO* are decommutated and defined into individual detector, gimbal position, and instrument engineering parameters or data elements. Once this is done, *Packet* acts as a server or distributer of these data elements to the other IPS classes. These other classes act upon the data elements either to convert them for subsequent processing, or to be stored for archival purposes. *Packet* uses the mandatory PCF, Time, and SMF Toolkit routines.

Instrument Configuration: This class corresponds to an abstraction of the instrument's mode and state that is based on a set of instrument parameters. *Instrument Configuration* serves as a distributer of instrument engineering data from *Packet* to the other instrument subassembly classes (Figure 2-2). *Instrument Configuration* also makes a determination of the overall health and status of the instrument based on the statuses of the individual instrument subassemblies. The resulting instrument configuration state is returned for other unrelated IPS class usage. *Instrument Configuration* and its subassembly classes use the mandatory PCF and SMF Toolkit routines.

The Instrument subassembly classes illustrated in Figure 2-2 are described below.

> *Azimuth Assembly*: converts and evaluates instrument engineering parameters related to the azimuth assembly and returns a status to *Instrument Configuration*.
>
> *Blackbody Assembly:* converts and evaluates instrument parameters related to the blackbody assembly and returns a status to *Instrument Configuration*.
>
> *Brake Assembly:* converts and evaluates instrument engineering parameters related to the brake assembly and returns a status to *Instrument Configuration*.
>
> *Commands:* evaluates a series of instrument command data and returns a status to *Instrument Configuration*.
>
> *Data Acquisition Assembly (DAA)*: converts and evaluates instrument engineering parameters related to the DAA and returns a status to *Instrument Configuration*.
>
> *Detector Assembly:* converts and evaluates instrument engineering parameters related to the detector assembly and returns a status to *Instrument Configuration*.
>
> *Elevation Assembly:* converts and evaluates instrument engineering parameters related to the elevation assembly and returns a status to *Instrument Configuration*.
>
> *Instrument Controller Assembly (ICA):* converts and evaluates instrument engineering parameters related to the ICA and returns a status to *Instrument Configuration*.

Main Cover Assembly: converts and evaluates instrument engineering parameters related to the main cover assembly and returns a status to *Instrument Configuration*.

MAM Assembly: converts and evaluates instrument engineering parameters related to the MAM assembly and returns a status to *Instrument Configuration*.

Power Converter Assembly (PCA): converts and evaluates instrument engineering parameters related to the PCA and returns a status to *Instrument Configuration*.

Solar Presence Sensor (SPS) Assembly: converts and evaluates instrument engineering parameters related to the SPS assembly and returns a status to *Instrument Configuration*.

SWICS Assembly: converts and evaluates instrument engineering parameters related to the SWICS assembly and returns a status to *Instrument Configuration*.

BiDirectional Scan (BDS) File: This class creates, writes, and closes the archivable BDS file and serves as a data receptor to other classes for raw and converted instrument detector and engineering data, calculated geolocation data, and quality assurance and control information. *BDS File* can cause *Controller* to halt IPS processing if a BDS file cannot be created or written. *BDS File* uses the mandatory PCF and SMF Toolkit routines. HDF routines are also used.

Instrument Earth Scan (IES) File: This class creates, writes, and closes the 24 1-hour IES files and is responsible for spatially sorting geolocated, filtered radiance footprints into these 1-hour files. *IES File* uses the mandatory PCF and SMF Toolkit routines. HDF routines are also used.

Location: This class combines instrument elevation and azimuth gimbal position data and spacecraft ephemeris position and velocity data to calculate geophysical data (latitude and longitude) for a given set of radiance values. Positional data used by the Cloud Properties Subsystem are also calculated and included in the *IES*. *Location* uses the mandatory PCF, Time, Ephemeris, Coordinate Conversion, and SMF Toolkit routines.

Radiance: This class converts instrument detector raw counts (shortwave, longwave, and total) to filtered radiance science values. QA on converted values is performed and quality flags for all converted radiance values are generated. *Radiance* uses the mandatory Time and SMF Toolkit routines.

Scan: This class acts as a base class for the following classes described below. *Scan* relies on input from *Controller and Instrument Configuration* to select the appropriate elevation scan profile class to be implemented for the current packet to be processed. These scan profile classes define various Instrument field-of-view data sets corresponding to the elevation gimbal position relative to anticipated viewing scenes. Typical scene data sets are categorized by Space Looks, Limb-to-Top-of-Atmosphere, Earth, Internal Calibration, and MAM Viewing Measurements. *Scan* and its subassembly classes use the mandatory PCF and SMF Toolkit routines.

Normal Earth Scan: defines the scan profile for the typical, normal science scan pattern while the instrument is performing an azimuth crosstrack or biaxial scan. Used by *Footprint*.

Short Earth Scan: defines the scan profile for a short scan pattern while the instrument is in an azimuth crosstrack or biaxial scan. Used by *Footprint*.

MAM Scan: defines the scan profile where the instrument views the Mirror Attenuator Mosaic Assembly and the internal calibration sources while in the solar calibration mode.

NADIR Scan: defines the scan profile for a NADIR scan by the instrument (detectors pointing down at Earth along the -Z axis).

Stowed Scan: defines a nonscanning profile when the instrument elevation gimbal is stowed for calibration, diagnostic, or other modes.

Footprint: This class coordinates the conversion of instrument detector measurement data into radiance values and geophysical coordinates. Provides the definition for a set of geolocated radiances for use by *IES*. Footprint uses the mandatory SMF Toolkit routines.

Additional classes that support this architecture (not shown, but used by most of the classes) include the following.

Run Time Parameters: This class provides the interface links between the Subsystem code and the run time environment. Parameters that are externally set and called by client classes include validation or debugging code invocation; level-0, BDS, IES, and ancillary file handlers; spacecraft and instrument identifiers; and start and stop times.

Log SMF: This class provides the interface links between the Subsystem code and the run time Toolkit error and process log calls. Client classes will provide and report all error messages via this class. Included in the class are calls to the Toolkit LogReport, LogStatus, and LogUser files. Error messages are further categorized by the following status levels: Success, Shell, Action, Message, User Info, Notice, Warning, Error, or Fatal. In addition, users can use this class as a means for reporting debugging or other useful diagnostic information.

System Types: This class provides basic system (global) parameter definitions to be used by all classes. These definitions are mapped directly from the problem domain. Examples include the packet sample number reference (e.g., 0-659) and packet range.

Base Types: This class provides common data (global) type definitions to be used by all classes. These definitions are used to provide consistent data processing results from one class to the next. This module is similar in concept to a FORTRAN data block.

Analog Parameters: This class provides the instrument engineering parameter definitions to be used by all classes and are mapped directly from the problem domain. These definitions are used to provide consistent parameter usage from one class to the next.

Digital Parameters: This class provides the instrument status parameter definitions to be used by all classes and are mapped directly from the problem domain. These definitions are used to provide consistent parameter usage from one class to the next.

Analog Conversion: This class supplies the equations necessary to convert instrument engineering counts to engineering unit data within a common module. Clients will provide the algorithm reference index and the raw count and will receive the converted value.

2.2 Scenario Diagrams

Scenario diagrams in the following sections are intended to define the dynamic behavior of the IPS as it processes data according to the primary functions. Scenarios described include:

- *Initialization Scenario*
- *Main Loop Processing Scenario*

The numbers in parenthesis within the scenario descriptions represent corresponding numbers shown in the Scenario Diagram figures.

2.2.1 Initialization Scenario

The initialization scenario for the IPS is shown in Figure 2-3. This scenario defines the events and processes that must occur within IPS before performing any data conversions, geolocations, verifications, or data storage. A detailed description of the flow of events between class objects is described below.

1. Upon start of a PGE execution, the *Controller* initializes *Level-0 IO* (1). In response, *Level-0 IO* attempts to retrieve the level-0 file handle from *Run Time Parameters (1.1)*. *Run Time Parameters* either returns the file handle or an error message. If a file handle is returned, *Level-0 IO* attempts to open the file and read the header. If the attempt fails, or the header is unreadable, or *Run Time Parameters* was unable to return the file handle, *Level-0 IO* sends an error message to *Controller* and processing is gracefully aborted. If the file is opened and the header successfully read, execution continues to the next initialization step.
2. *Controller* sends an initialization message to the *BDS File* object (2). Similar to above, the *BDS File* object sends a message to *Run Time Parameters* requesting the BDS file name (2.1). If a file name is retrieved, *BDS File* attempts to create the new file. If unsuccessful, or if *Run Time Parameters* was unable to return a valid BDS file name, an error message is sent to *Controller* from *BDS File* and processing is aborted. Otherwise, execution continues.
3. *Controller* sends an initialization message to the *Instrument Configuration* (3). Similar to above, the *Instrument Configuration* object sends a message to the subassembly classes. For each subassembly class that require supporting data files, they call *Run Time Parameters* requesting the appropriate file handle (3.1). If the file name is retrievable, the subassembly class attempts to either open the existing file or create a new file. If

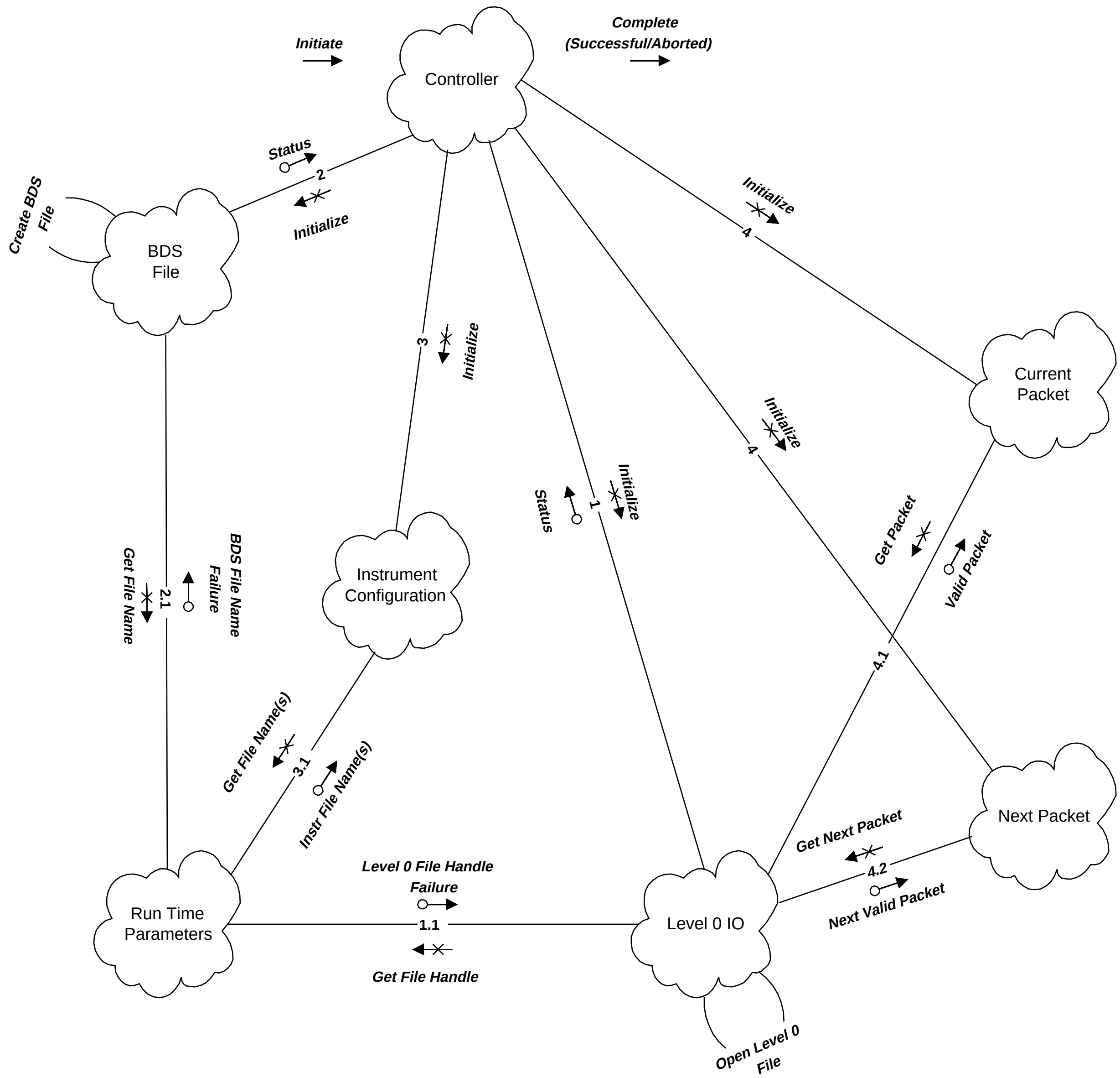

Figure 2-3. Scenario Diagram: Initialization Scenario

unsuccessful or if *Run Time Parameters* was unable to return a valid file name, an error message is sent to *Instrument Configuration*, in turn to *Controller*, where processing is aborted. Otherwise, execution continues.

4. In the final sequences of the Initialization scenario, *Controller* calls *Packet* (4) to initialize two Packet objects, *Current Packet* and *Next Packet*. Both Packet objects use the *Level-0 IO* object to retrieve valid packets from the level-0 file. As their names imply, *Current Packet* initializes with the first valid level-0 file packet (4.1), while *Next Packet* initializes with the subsequent valid packet (4.2).

2.2.2 Main Loop Processing Scenario

Upon completion of the initialization sequence, assuming that the IPS processing was not prematurely terminated, the main processing loop sequence begins. This iterative process reads sequential packets from the level-0 file, validates the packet header data, converts and analyzes the instrument and science data, and writes the output BDS and IES data products.

The scenario diagram in Figure 2-4 details this sequence of events for a single iteration of the IPS main processing loop. A detailed description of the flow of events between objects is provided below.

1. Following initialization, the *Current Packet* and *Next Packet* objects decompose their respective packet data into constituent parts in order to act as data servers for other system objects (1, 1.1). Raw science, analog, and digital data is then written to the BDS File (1.2).
2. *Controller* requests instrument status and scan mode data from the *Instrument Configuration* object (2). *Instrument Configuration* responds by getting unconverted instrument data (analog and digital data) from *Current Packet* (2.1). *Instrument Configuration* converts the raw data (2.2), returns instrument scan mode and status to *Controller*, and then writes converted data to the *BDS File* (2.3).
3. *Controller* passes instrument scan data to the *Scan* object (3). *Scan* uses this data to select the appropriate type of elevation scan (Normal Earth, Short, NADIR, etc.) profile to use for processing (3.1).
4. *Footprint* uses scan profile data from one of the *Scan* objects to determine subsequent science processing requirements (4).
   a) *Footprint* begins processing science data by retrieving unconverted detector and geolocation data from *Current Packet* and *Next Packet* (4.1.1, 4.1.2).
   b) *Footprint* uses *Location* to convert geolocation data (satellite ephemeris) to their corresponding geophysical units (4.2). *Location* in turn, requires some parameters (FOV elevation and azimuth positions) from *Instrument Configuration* in order to geolocate the footprint (4.2.1).
   c) Depending on the scan profile, some *Footprint* space look detector samples from both the current and next packet will be used by *Space Clamp* to compute parameters that support radiometric conversions (4.3).
   d) *Radiance* is used to convert all *Footprint* detector counts from the current packet (4.4). *Radiance* in turn, requires some parameters from *Instrument Configuration* (4.4.1) and *Space Clamp* (4.4.2) for radiometric conversion.
5. *Footprint* writes all converted radiance and location values to the *BDS File* (5).

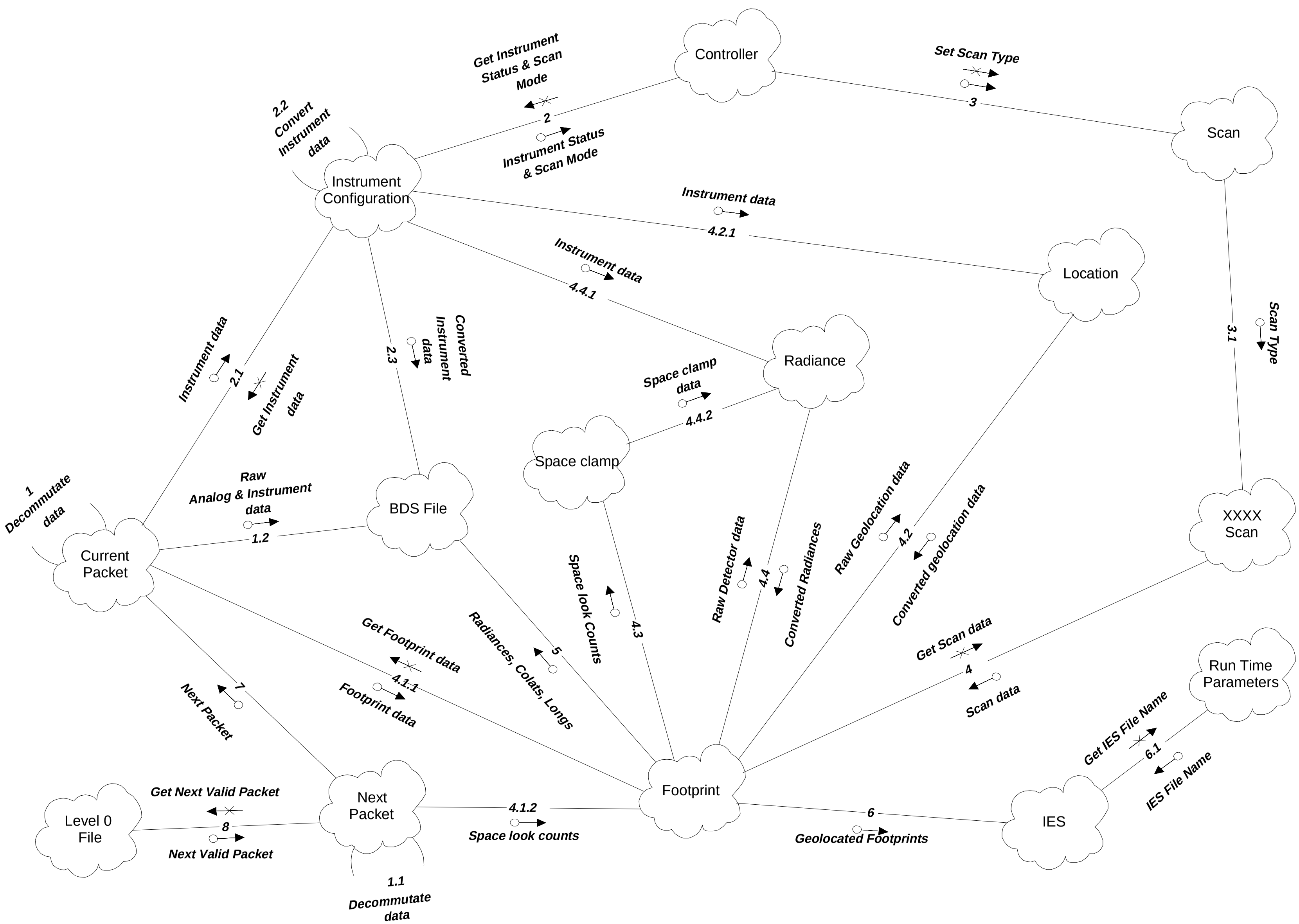

Figure 2-4. Scenario Diagram: Main Loop Processing Scenario

6. *Footprint* writes all resulting Earth-viewing-only converted radiance and location values to the *IES* (6). *IES* stores up to 80 minutes of the geolocated radiance data as data are received. At the end of 80 minutes, *IES* will use *Run Time Parameters* (6.1) to retrieve the file names needed to create the corresponding 1-hour IES output file, perform spatial sorting, write data to the external file, and clear any buffers for subsequent data.
7. Finally, once all of the data samples in the current packet have been converted and written, *Next Packet* moves the raw data from *Next Packet* to *Current Packet* (7), retrieves the next packet from *Level-0 IO* (8), and the process repeats until all packets in the level-0 file have been processed.

Additional detailed main loop processing scenarios are expected to be developed for the following subprocesses. These will be described in the detailed design document.

- *Level-0 File Verification*
- *Packet Retrieval and Processing*
- *Instrument Configuration Processing*
- *Geolocation and Ephemeris Verification Processing*
- *Space Clamp and Radiance Processing*
- *Solar Calibration Processing*
- *Hierarchical Data Format Processing*
- *QC and Report Processing*
- *Instrument Diagnostic Data Processing*

References

1. CERES Algorithm Theoretical Basis Document (ATBD), Instrument Geolocate and Calibrate Earth Radiances (Subsystem 1.0), Release 1.1, April 1994.
2. CERES Data Management System Software Requirements Document (SRD), January 1995.
3. CERES Data Management System Interface Requirements Document (IRD), September 1994.
4. CERES Data Management System Data Products Catalog (DPC), August 1994.
5. Specifications of Algorithms to Calculate Geolocations of the CERES Instrument Radiance Measurements and other Earth-Sun-Spacecraft Parameters for the CERES DMS, December 1995, Version 1.01.
6. CERES Instrument Operations Manual, November 1993.
7. CERES In-flight Measurement Analysis, DRL-64, September 1995.
8. Calibration Concepts for the CERES Proto-Flight Model (PFM) on the Tropical Rainfall Measurement Mission (TRMM), rev. 6, August 1995.
9. ECS Operations Concept Document for the ECS Project, August 1993.
10. Science Data Production Toolkit User's Guide for the ECS Project, August 1995.
11. Ada Quality and Style: Guidelines for Professional Programmers, Version 2.01, December 1992.

Appendix A

Abbreviations and Acronyms

Appendix A
Abbreviations and Acronyms

API	Application Programming Interface
APID	Application Identifier
ATBD	Algorithm Theoretical Basis Document
BDS	BiDirectional Scan
CCSDS	Consultative Committee for Space Data Systems
CERES	Clouds and the Earth’s Radiant Energy System
DAA	Data Acquisition Assembly
DAAC	Distributed Active Archive Center
DMS	Data Management System
ECS	EOSDIS Core System
EGMOD	Earth Geoid Model
EOS	Earth Observing System
EOS-AM	EOS Morning Crossing Mission
EOS-PM	EOS Afternoon Crossing Mission
EOSDIS	Earth Observing System Data and Information System
ERBE	Earth Radiation Budget Experiment
ERBS	Earth Radiation Budget Satellite
FOV	Field-of-View
HDF	Hierarchical Data Format
ICA	Instrument Control Assembly
ICCOEF	Instrument Calibration Coefficients
IENGCOEF	Instrument Engineering Coefficients
IES	Instrument Earth Scan
INSTR	Instrument
IPFCOEF	Instrument Platform Coefficients
IPPR	Instrument Production Process QC Reports
IPS	Instrument Processing Subsystem
IRCOEF	Instrument Radiance Coefficients
ITREND	Instrument Trends
IUSER	Instrument User
MAM	Mirror Attenuator Mosaic
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration

OODA	Object Oriented Design and Analysis
PCA	Power Converter Assembly
PFM	Proto-Flight Model
PGE	Product Generation Executive
PGS	Product Generation System
QA	Quality Assurance
QC	Quality Control
SCF	Science Computing Facility
SMF	System Management Function
SPS	Solar Presence Sensor
SPSEN	Spectral Sensitivity
SWICS	Shortwave Internal Calibration Source
TDRSS	Tracking and Data Relay Satellite System
TOA	Top-of-Atmosphere
TRMM	Tropical Rainfall Measurement Mission